multiple single language dictionaries. It can also be comprised of such things as sequences of nucleotides in biology, or any collection of valid "words" consisting of letters from a pre-established "alphabet." While the dictionary size and alphabet size are presumed to be large, their actual size is unimportant, and average/maximum word length is assumed to be relatively small, i.e., many orders of magnitude smaller than the dictionary size.

[0017] For a dictionary size, D, alphabet size, A, and a maximum word length, W, the disclosed algorithm corrects distance one misspellings in O(W) time and distance two misspellings in O(W^2) time. The required storage is O(D), or in the case W varies with D, equal to O(D*W), for distance one misspellings and O(D*W^2) for distance two misspellings. In general, it is assumed that W is more or less constant and does not grow with D so that O(D) equals O(D*W) or O(D*W^2).

[0018] According to a further aspect of the invention a soft algorithm is disclosed that uses a "soft" definition of distance two misspellings, where distance two spelling correction can be performed in O(W) time and O(D)=O(D*W) storage. "Soft" distance two means that only the following distance two errors are considered: double transposition, transposition-deletion, transposition-insertion, deletion-transposition, deletion-insertion, insertion-transposition, and insertion-deletion.

[0019] FIG. **1** is a flow chart illustrating the overall flow of an exemplary distance one spelling correction algorithm **100**. In some instances of the spelling correction problem, it is adequate to detect only distance one spelling errors, and furthermore, distance one detection is the first step in the various distance two correction algorithms. A wild card is an arbitrary symbol, indicating a wildcard that is assumed to not appear in any dictionary word.

[0020] In the following discussion, the verb "to hash" or any of its grammatical variants refer to the act of placing something in a hash table. For example, the phrase "hashing all dictionary words" means placing all dictionary words in a hash table. Uses of hashtables and performance guarantees for simple hash table operations such as insertion and lookup are described in any standard reference on algorithms. See, for example, C. Cormen et al., *Introduction to Algorithms*, MIT Press (2001).

[0021] The method involves hashing all dictionary words, in a known manner, and all "replacements" of dictionary words, in accordance with the present invention. Replacements are hashed, using, for example, an asterisk '*' as a wild card, as follows. If the dictionary word is COAT, then the following variants are hashed: *OAT, C*AT, CO*T and COA*. In general, if a word is of length W, then W such word variants are hashed. The (key, value) pairs are (*OAT, COAT), (C*AT, COAT), (CO*T, COAT), and (COA*, COAT). Separate hash tables are kept for the words (i.e., the dictionary) and for the replacement variants. These hash tables are assumed to be pre-created prior to when the distance one spelling corrector starts up (Step **110**).

[0022] In response to obtaining the input candidate word (Step **120**), say in this case the term is WXYZ, one first checks the word against the direct dictionary hash (Step **130**). One then gets to the decision point **140**. If a match is found in the dictionary hash, then the word is spelled correctly, and the program terminates indicating the correct spelling, as in Step **150**. If, however, no match is found, a misspelling is assumed and one checks all distance one variants against the appropriate distance one hash tables, accumulating suggested spelling corrections using the process **200**, discussed further below in conjunction with FIG. **2**. Finally, in Step **160** the suggested corrections are output.

[0023] FIG. **2** is a flow chart illustrating an exemplary process **200** of testing variants of the candidate word against hash tables derived from the dictionary for distance one misspellings in accordance with the present invention. Upon starting and obtaining the candidate word (Step **210**), one first generates all transpositions of adjacent characters, and single character deletions of the candidate word (Step **220**). For the candidate word WXYZ, the transpositions would be XWYZ, WYXZ, and WXZY. The deletions would be XYZ, WYZ, WXZ, and WXY. These are each checked against the dictionary hash in Step **230** and any matches are accumulated. The transposition checking will undo a misspelling of the same kind since the inverse of a transposition is the same transposition and the deletion checking will undo a corresponding insertion. The next step is to generate all single character replacements and insertions (Step **240**) and test these against the replacement hash (Step **250**). Replacements in this case are *XYZ, W*YZ, WX*Z, and WXY*. Insertions are *WXYZ, W*XYZ, WX*YZ, WXY*Z, and WXYZ*. Replacements catch distance one replacement errors, and insertions catch distance one deletion errors. As usual, the final step is to output all hash table matches (Step **260**). The total effort expended is 4W hash lookups which is O(W), and the memory used for storage of the hash is O(D)=O(D*W).

[0024] This algorithm affords no false positives. In other words, the algorithm never suggests a spelling correction that is more than distance one from the original word. On the other hand, if one were to just hash the dictionary together with all ordered subsequences of dictionary words of length W−1 as in Greene et al., "Multi-Index Hashing for Information Retrieval," 35th Annual Symposium on Foundations of Computer Science, 722-731 (1994), and do a corresponding lookup, one would run into false positives. For example, for both the dictionary words COAT and OATH the ordered subsequence OAT would be hashed, and both would be a suggested distance one correction in response to the query "DOAT," despite the fact that OATH is not distance one from DOAT.

[0025] FIG. **3** is a flow chart illustrating the overall flow of the distance two spelling correction algorithm **300**. The method **300** involves the utilization of certain hash tables that are assumed to be pre-created. The following hash tables are needed: a transposition (t) hash, a deletion (d) hash, a transposition-replacement (tr) hash, a deletion-transposition (dt) hash, a double deletion (dd) hash, a deletion-replacement (dr) hash, and an insertion-replacement (ir) hash. Only a special form of the deletion-transposition hash is required, namely, those deletions followed by transpositions that first delete a character, and then transpose the characters initially surrounding the deleted character, as discussed further below. Each of these hash tables contains keys that correspond to certain variant forms of each dictionary word. The contents of each hash are again illustrated by considering the sample dictionary word COAT. Although the contents of the hash table are (key, value) pairs, in all cases for the dictionary word COAT, value ═COAT, so only the keys are shown.